ASSIGNMENT 16.1

Task 1
Create a calculator to work with rational numbers.
Requirements:
➢ It should provide capability to add, subtract, divide and multiply rational Numbers
➢ Create a method to compute GCD (this will come in handy during operations on rational)
Add option to work with whole numbers which are also rational numbers i.e. (n/1)
➢ achieve the above using auxiliary constructors
➢ enable method overloading to enable each function to work with numbers and rational.

SOLUTION
The complete source code has been given below. Following are the main properties of the program

1. It uses Rational as the main Scala class. It uses Rational companion Scala object to provide helper methods 'apply' to avoid creating new instances of Rational class from outside. It makes the code look elegant
2. The constructor of the Rational class has been made private so that they can't be instantiated from outside and are only accessible via Rational Object
3. The Rational class primary constructor takes both numerator and denominator
4. The auxiliary class takes only one parameter(numerator) and calls the primary constructor with denominator as 1
5. The method name is same as the operator, (+, -, *, /) which is allowed in scala
6. There is no need for method overloading as apply overloading is sufficient for this purpose to handle cases of Rational numbers and integers
7. The program prints the output on the console
8. The code has relevant comments and the required places
9. Following is the way various Rational operations are performed in the code

```
// fraction addition
Rational(90,31) + Rational(34,53)
// fraction subtraction
Rational(2,31) - Rational(34,53)
// fraction multiplication
Rational(2,5) * Rational(9,6)
// fraction division
Rational(2,5) / Rational(9,6)
// integer addition
Rational(2) + Rational(9)
// integer subtraction
Rational(100) - Rational(2)
// integer multiplication
Rational(25) * Rational(4)
// integer division
Rational(100) / Rational(25)
```

10. Following is the output of the above

```
90/31 + 34/53 = 5824
2/31 - 34/53 = -948
2/5 x 9/6 = 3/5
2/5 / 9/6 = 4/15
2 + 9 = 11
100 - 2 = 98
25 x 4 = 100
100 / 25 = 4

Process finished with exit code 0
```

SOURCE CODE

```
object Rational {
  // companion object to Rational
  def main(args: Array[String]): Unit = {

    // fraction addition
    Rational(90,31) + Rational(34,53)
    // fraction subtraction
    Rational(2,31) - Rational(34,53)
    // fraction multiplication
    Rational(2,5) * Rational(9,6)
    // fraction division
    Rational(2,5) / Rational(9,6)
    // integer addition
    Rational(2) + Rational(9)
    // integer subtraction
    Rational(100) - Rational(2)
    // integer multiplication
    Rational(25) * Rational(4)
    // integer division
    Rational(100) / Rational(25)


  }

  // helper functions to instantiate Rational class for both rational and integers
  def apply(n1: Int, d1: Int)={
    new Rational(n1,d1)
  }
  def apply(x: Int)={
    new Rational(x)
  }


}
```

```scala
 class Rational private(val n1: Int, val d1: Int) // primary constructor, private
{

  private def this(x:Int)= // auxiliary constructor, private
  {
    this(x,1)
  }


  def +(r: Rational): Unit = {

    // cross multiply the fractions and then add which becomes the result numerator
    // multiply denominators which becomes result denominator
    // get the greatest common divisor of resulting numerator and denominator
    // and divide resultant numerator and denominator with it
     val n = n1*r.d1 + d1*r.n1
     val d = d1*r.d1
     val gcd = GCD(n,d)
     println(frp(n1,d1) + " + " + frp(r.n1,r.d1) + " = "+ frp(n,gcd))

  }

  def -(r: Rational): Unit = {
    // cross multiply the fractions and subtract which becomes the result numerator
    // multiply denominators which becomes result denominator
    // get the greatest common divisor of resulting numerator and denominator
    // and divide resultant numerator and denominator with it
    val n = n1*r.d1 - d1*r.n1
    val d = d1*r.d1
    val gcd = GCD(n,d)
    println(frp(n1,d1) + " - " + frp(r.n1,r.d1) + " = "+ frp(n,gcd))

  }

  def *(r: Rational): Unit = {
    // multiply both numerators which becomes the resultant numerator
    // multiply both denominators which becomes the resultant denominator
    // get the greatest common divisor of resulting numerator and denominator
    // and divide resultant numerator and denominator with it
    val n = n1*r.n1
    val d = d1*r.d1
    val gcd = GCD(n,d)
    println(frp(n1,d1) + " x " + frp(r.n1,r.d1) + " = "+ frp(n/gcd,d/gcd))


  }

  def /(r: Rational): Unit = {
    // logic is same as multiple but invert the second rational,
    // basically cross multiply the fractions
    val n = n1*r.d1
    val d = d1*r.n1
    val gcd = GCD(n,d)
    //println(n1+"/"+d1+" / "+r.n1+"/"+r.d1+" = "+ n/gcd+"/"+d/gcd)
    println(frp(n1,d1) + " / " + frp(r.n1,r.d1) + " = "+ frp(n/gcd,d/gcd))
  }
// helper function to find greatest common divisor

  private def GCD(x: Int, y: Int): Int = {

    if (y == 0) x
    else
      GCD(y, x % y)
  }
  // helper method to print fraction or as integer if the denominator is 1
  private def frp(n:Int,d:Int):String=if(d==1) n.toString else n+"/"+d

}
```